

IMAGE PROCESSOR

Patent Number: JP1311379
Publication date: 1989-12-15
Inventor(s): KAWAMURA NAOTO
Applicant(s): CANON INC
Requested Patent: ☐ JP1311379
Application Number: JP19880141825 19880610
Priority Number(s):
IPC Classification: G06F15/66; G06F15/72; H04N1/21; H04N1/411
EC Classification:
Equivalents: JP2787830B2

Abstract

PURPOSE:To allow an image memory with small capacity to execute processing by developing and compressing a vector image corresponding to a partial image on an image memory having the capacity of the partial image and storing the processed image correspondingly to the whole image.

CONSTITUTION:A real address space 4K X 1K of an image relating to a formed graphic data stored in a work area or the like of a CPU 51 is segmented to 1K X 1K of 16 block units in total and one, e.g. a block A00, of them is extracted to a source image memory 44, its corresponding linear image is drawn, the image is compressed by a compressor 115, and the compressed image is stored in a block B00 on a corresponding destination image memory 40 as a compression code. Thereafter, similar processing is successively applied to 1K X 1K blocks in the real address space and compressed highly accurate images are stored in the image memory 40, so that images can be processed by the image memory with small capacity.

Data supplied from the esp@cenet database - I2

⑫ 公開特許公報(A)

平1-311379

⑤ Int. Cl. 4

識別記号

庁内整理番号

④ 公開 平成1年(1989)12月15日

G 06 F 15/66
15/72
H 04 N 1/21
1/411

3 3 0
3 5 0

H-8419-5B
8125-5B
8839-5C
7060-5C

審査請求 未請求 請求項の数 1 (全14頁)

⑥ 発明の名称 画像処理装置

⑦ 特 願 昭63-141825

⑧ 出 願 昭63(1988)6月10日

⑨ 発 明 者 河 村 尚 登 東京都大田区下丸子3丁目30番2号 キヤノン株式会社内
⑩ 出 願 人 キヤノン株式会社 東京都大田区下丸子3丁目30番2号
⑪ 代 理 人 弁理士 大塚 康徳 外1名

明 細 書

1. 発明の名称

画像処理装置

2. 特許請求の範囲

少なくともベクトル画像の部分画像分の容量を
有する画像メモリと、

該画像メモリ内で前記部分画像に対応するベク
トル画像を展開する展開手段と、

該展開手段で展開された部分ベクトル画像を圧
縮する圧縮手段と、

圧縮された部分ベクトル画像を前記全体画像に
対応付けて記憶する記憶手段とを備え、

前記ベクトル画像に対する全ての圧縮部分ベク
トル画像を前記記憶手段で記憶することを特徴と
する画像処理装置。

3. 発明の詳細な説明

〔産業上の利用分野〕

本発明は画像処理装置、特に小容量の画像メモ
リで大きい画像の処理を行う画像処理装置に関す
るものである。

〔従来の技術〕

近年では、画像処理技術の発展に伴ない、画像
の表示或いはプリント出力時における画像の高細
精化が求められており、このために表示画面への
表示或いは印刷出力時にかかる画像メモリの容量
は膨大になることが考えられる。

例えばA4サイズに400dpiの画像を出力し
ようとすれば画素数で約16,000,000画
素にもなり、画像データとして8bit/画像で持
つとするとA4ページ分で約16メガ・バイト、
カラーの時R, G, B各8bitずつ持たせるとす

れば約48メガ・バイトの画像メモリを必要とし、装置の大型化、コスト・アップは避けられないし、通信にも時間がかかる。

〔発明が解決しようとする課題〕

そこで、原画像に対して、例えば3×3画素を1表示画素まで圧縮して格納することが考えられるが、この様な形態で保持されているメモリ上でベクトル画像を展開しようとする、以下に示す問題が発生する。

すなわち、圧縮データを複号し、そして所定のドットを書込み、再び符号化（圧縮）するという処理を、その1つベクトル画像の全てのドットに対して行わねばならず、装置の処理速度は著しく低下する。

また、個々のベクトル画像を展開するときに上述した処理を繰り返していくことになるので、次

像を前記全体画像に対応付けて記憶する記憶手段とを備える。

〔作用〕

かかる本発明の構成において、部分画像の容量を有する画像メモリ上でその部分画像に対応するベクトル画像を展開する。そして、展開されたベクトル画像を圧縮して、圧縮された部分画像を全体画像に対応付けて記憶していくものである。

〔実施例〕

以下本発明の実施例に沿って説明する。

<基本構成の説明（第1図）>

まず、本実施例における画像編集装置の構成を第1図に示し、その動作概略を以下に説明する。

リーダ10から読み取られた画像データは、圧縮器115により圧縮され、画像メモリ40或いは41へ汎用のSCSIインタフェース20を通

第に画像自体が劣化する。

本発明はかかる課題に鑑みなされたものであり、増大する画像データを圧縮することで小容量の画像メモリで処理することを可能とすると共に、ベクトル画像の編集における画像劣化を防ぎつつ画像劣化を抑えることを可能ならしめた画像処理装置を提供しようとするものである。

〔課題を解決するための手段〕

この課題を解決するために、本発明は以下に示す構成を備える。

すなわち、

少なくともベクトル画像の部分画像分の容量を有する画像メモリと、該画像メモリ内で前記部分画像に対応するベクトル画像を展開する展開手段と、該展開手段で展開された部分ベクトル画像を圧縮する圧縮手段と、圧縮された部分ベクトル画

して取り込まれる。画像メモリ40、41の内容はルックアップテーブル（以下、単にLUTという）32、33により色変換処理された後D/Aコンバータ36a、b、cによりアナログのR、G、B信号に戻され、モニタ装置37に表示される。

この時、編集の機能を高めるために実施例における画像メモリはソース用画像メモリ44、中間バッファ用画像メモリ41、及びディスプレイネーション用画像メモリ40の3つを有し、各々1頁分の容量を持つ。各メモリは2つのバス（CPUバス30と、サブCPUバス31）とに選択的に接続されている。かかる選択は、メインCPU50からの指令でCPUバス30内に存在するコントロールBUSを介して行われる。即ち、各画像メモリ40、41、44はメインCPU50の指

令によりCPUバス30、及びサブCPUバス31へ常時1つずつ接続されている。モニタ装置37へは2つの画像メモリ40、及び41がマスクメモリ42の情報によつて合成されて表示される。ここで、マスクメモリ42の情報が、画像メモリ40、41と同期してビデオレートで読み出されており、マスクメモリ42の内容により画像メモリ40を表示するか、画像メモリ41を表示するかを決定する。従つて、マスクメモリ42の出力で合成器34にて2つの画像メモリの読取りデータを選択する事により、2つの画像の合成がモニタ37上で成されることになる。

<画像データの圧縮の説明>

(第2図～第10図)>

次に画像データの圧縮について述べる。

第2図は原画を画素単位の構成で示した図であ

として4ビット、明度差1、2としてそれぞれ3ビット、色度差1、2としてそれぞれ4ビットより成るものである。尚、原画の各画素は各色情報であるR、G、Bが6ビット、合計18ビットのカラー情報をもつデータである。従つて4×4画素では合計288ビット(=18×16)となり、1/9に圧縮することになる。

第6図は原画の各R、G、Bデータ6ビットから圧縮データを作成し、メモリに格納する圧縮器115のブロック構成図である。

図中、61は18ビット入力18ビット出力のルックアップテーブルであり、62～64はそれぞれ4本のラインバッファが2組ずつよりなるトグルバッファである。また、65は明度データより明度平均及び4×4画素を明度平均よりも値の大きな画素領域と小さな画素領域に分け(2値

り、第3図は原画の4×4画素領域を単位(以下、この画素領域をスーパー画素と称する)として圧縮し、圧縮後のデータをスーパー画素データとして、スーパー画素単位で構成した状態を示す図である。また、第4図は原画の4×4画素領域からスーパー画素の対応を示す図である。

本実施例では高精細な画像を圧縮された形で各メモリに保持し、その圧縮されたデータをディスプレイ上で表示しながら編集作業を行い得るようにし、これにより、メモリコスト及びディスプレイ装置コストを控えようとするものである。

また、第5図は圧縮されたデータの一例を示したものである。データ長(各スーパー画素)は32ビットで構成されており、原画の4×4画素よりなる領域の明度平均値として6ビット、色度平均値として8ビット、またブロックパターン情報

化)、その領域の境界情報をブロックパターン情報として出力する検出部である。また検出部65では境界分けされた各画素領域での明度平均と前述した4×4画素領域明度平均との差情報を明度差1及び明度差2として出力するものである。ここで、明度差1は、平均よりも大きな値をもつ領域の差情報であり、明度差2は平均よりも小さな値をもつ領域の差情報を意味する。また、明度データとは均等色空間 $L^*a^*b^*$ で表現される L^* を用いて算出され、以降説明する色度情報に関しては a^*b^* を用いて算出される。原画のR、G、Bデータから $L^*a^*b^*$ にはLUT 61を用いて変換されるものである。

以降圧縮データの作成法に関し、更に詳細に説明する。

第7図はトグルバッファ62～64の構成を詳

細に説明するための図である。

図中、74-1～74-4及び74-5～74-8はバッファメモリであり、バッファメモリペアをなしている（尚、以下の説明でバッファメモリ74-1～74-8をバッファ群77という。）。

また、各々のバッファメモリへのデータ入力はマルチプレクサ73-1、及び73-2により切り換えて用いられる。更にマルチプレクサ72は、入力した画像データの送り先をバッファ群76にするか或いはバッファ群77の出力をセレクトしてデータを送出するものであり、4本のバッファメモリからの出力をバラレルに出力するものである。カウンタ71は、マルチプレクサ72、73-1、73-2及びセクタ75を制御するためのもので、ラスタ同期信号をカウントするこ

を得るものである。

第9図はブロック平均算出回路81の一例であるブロック図である。

ここで4ラスタのデータがそれぞれ並列に入力されることにより16画素（4×4画素）分のデータを入力する。

先ず加算器91-1及び91-2は各々4画素分並列に入力されるデータを2画素ずつ加算する。バッファ92-1及び92-2はそれぞれ加算器91-1及び91-2の出力を一時保存する。次に加算器93-2では各々加算器91-1及び91-2の相連続する2出力の加算を行う。またバッファ94-1及び94-2では加算器93-1及び93-2の出力を一時保持する。更に加算器95-1及び95-2は、加算器93-1及び93-2の相連続する2出力の加算を行い、最後に

とにより、4ラスタ毎にマルチプレクサ72及びセクタ75をそれぞれ切り換え、更に1ラスタ毎にマルチプレクサ73-1（及びマルチプレクサ73-2）の出力先（バッファメモリ）を切り換えるものである。

第8図はL*の4ラインのデータから明度情報を出力する第6図の検出部65を説明する図である。図中、81はブロック内L*の平均を算出するブロック平均算出回路であり、82はブロック平均算出回路81がブロック内平均L*を算出するのに要する時間だけ遅延させるための遅延回路である。また84はブロック内の各画素のL*データとブロック平均算出回路81からの出力とから、ブロック内の領域分けと該各領域の領域平均とブロック平均との差分値を算出する回路である。これらにより、第5図でいう明度差1及び2

平均算出器96で加算器95-1及び95-2の出力の加算を行って、平均を計算するわけである。ここで、加算器93-1及び93-2は加算器91-1及び91-2の加算の周期の倍の周期で動作し、加算器95-1、95-2及び平均算出器96は更にその倍の周期で動作する。従って、加算器91-1及び91-2が4回加算動作する時に、加算器93-1及び93-2は2回動作し、加算器95-1、95-2及び平均算出器及びバッファを接続するバス上の数値は、そのバスのビット数を表わしている。

第10図は第8図に示した領域分け、領域平均、領域差分を算出する領域分け・領域平均・領域差分算出回路84の詳細である。

図中、100は第8図に示した遅延回路82より当該ブロック内の各画素のデータを入力し、ブ

ブロック平均算出回路81よりブロック平均を入力し各画素がブロック平均よりも大きな値をもつ領域にあるか、それともブロック平均以下の値をもつ領域にあるかを判定する比較器である。1ブロックは4ラスタで、かつ各ラスタは4画素で構成されるため、各ラスタに1個ずつ比較器100を持つものとして説明する。換言すれば、第10図の比較器100及びその周辺のカウンタ等は1ラスタに対して処理するものである。

さて、各比較器100の出力は、それぞれ2つのゲート101-1及び101-2の切換え信号として出力されており、ゲート101-1、101-2は共に同一の画素データを入力している。すなわち、比較器100の出力によりゲート101-1及び101-2のうちのいずれか一方は入力データをそのまま出力し、残りの一方は“0”

値)内の各画素の値の総和をとり、平均器104-2は、それぞれ領域1及び領域2の属する画素数及びその総和を入力し、平均値を出力するものである。また、加算器103-1及び103-2は、第9図で示す回路と同様な回路で構成できる。96の加算器の出力を、以下ビットを省略せずに出力するようにとる。差分器105-1及び105-2は、それぞれ領域1及び領域2の平均とブロック平均との差を出力するものである。またブロックパターンテーブル107は、各ラスタ4ビットずつのパターンを入力し、合計16ビットのパターンをパターンコードとして出力するものである。尚、平均器104-1、104-2及び差分器105-1、105-2、ブロックパターンテーブル107はROMのLUT(ルックアップテーブル)で容易に実現できる。

を出力するわけである。カウンタ102は比較器100の結果をカウントし、4画素中何画素が領域1(ブロック平均よりも大きな値をもつ領域)に入るかをカウントする。また、シフトレジスタ106は比較器の出力(2値)をシフトし、シフト状態をパラレルに出力するものである。これにより、ブロック内の領域1及び領域2(ブロック平均値以下の値を持つ領域)のブロックのパターンを表現するものである。以上の破線で囲まれた領域109は、各ラスタにそれぞれ存在する。

さて、加算器108では各ラスタの領域1内の画素数を全て加算し、平均器104-1及び104-2に対し、当該ブロック内の画素数を出力する。また、加算器103-1及び103-2は、各々ブロック内の領域1内の各画素、領域2(ブロック平均に等しいか、もしくはそれより小さい

色度に関しても全く同様に構成できるが、領域1、領域2の切り分けの信号に関しては、明度の場合のデータを用いて行い、データは色度データ a^* 及び b^* で行うものである。 a^* 、 b^* 独立にデータは扱われるが、 a^* 、 b^* を合わせて色度データとする。

かくして得られた圧縮データを前述の如く圧縮して表示データとして用いることにより、例えばA4サイズの原稿を16画素/mmの画素密度で読み取つたとすると、 4752×3360 画素のデータ量となるところを、 1188×840 スーパー画素とする事ができる。これにより1/9の圧縮率を得る。

前述の圧縮法により、第2図の画像データの画素配列は、第3図のスーパー画素配列は変換され小容量化される。例えば、A4判1ページ分で約

4M(メガ)バイト(キ1188×840×32ビット)相当となる。これはICメモリで容易に実用化でき、第1図における各画像メモリ40、41、43は各々この容量から成るものである。

以上の処理で画像メモリ40に格納された圧縮画像(仮に画像Aとする)と、画像メモリ41に格納された圧縮画像(仮にBとする)とを合成表示するときには、マスクメモリ42内のデータに従って、画像Aと画像Bとを合成器34で切換えてモニタ装置37に表示することになる。このとき、画像A及び画像Bは先(第5図)に説明した様にデータ長が32ビットとなっているが、表示する場合には明度平均と色度平均とを出力するだけで、他のブロックパターン等は印刷時等に復号するものである。

ここで、印刷時にはデータ長32ビット内のプ

る画像データを小容量化した。

ところで、画像データがグラフィックスデータで構成されたCADやGraphics及びoutline フォントとよばれるベクトルフォント等(例えば線画等)のものであると圧縮符号化の変換が複雑化するという欠点がある。

第11図(a)～(d)を用いて、この欠点を簡単に説明する。

今、同図(a)の如く、画像メモリ上へ2点A(x_1, y_1)、B(x_2, y_2)を結ぶ線分ベクトルABを描くとする。

画像メモリが非圧縮で通常のメモリであれば2点A、Bを結ぶ直線の方程式は

$$y = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1) + y_1$$

(但し $x_1 \leq x \leq x_2$)

ロックパターン(ここにはコードが格納されている)から一旦、4×4の2値化ブロックに復号し、“1”となつている画素に対する明度値を明度平均に明度差1を加えた値にし、色度値を色度平均に色度差1を加えた値にして復号する。

一方、“0”となつている画素に対してであるが、明度値を明度平均から明度差2を引いた値にし、色度差を色度平均から色度差2を引いた値にして復号することになる。

尚、これらの処理はルックアップテーブルを用いれば容易に達成される。

<ベクトル画像の説明(第11図～第15図)>

以上の様に本実施例においては、その対象とする画像は自然画像(リーダ10等からの画像)に対するものであつた。そして、2次元画像データをラスタ走査により圧縮コード化し大容量化する

で求め、整数化した(x, y)アドレスヘラインの輝度(又は濃度)情報や色情報を書き込めばよい。

一方、画像メモリが前述の圧縮法による、いわゆるスーパー画素単位のメモリであれば同図(b)に示す様に4×4の画素から成るスーパー画素内での描画情報を圧縮器115へ送り符号化して画像メモリーへ送る必要がある。この時、同図(c)の様に既に描かれた線分ベクトルABの上に重ねて線分ベクトルCDを描く時その交点E(110)に於て下地の符号データから重なった線分を表わす符号データは変換する必要がある(同図(d))。即ち、交点E(110)に於てはベクトルABを描画した時点で既に符号化されており、その上へ別のベクトルCDを重ね書きする時には以下の処理を踏むことになる。

①交点Eを含むスーパー画素データを復号化し元の 4×4 のデータに戻す。

②重ね ぎするデータをこの 4×4 のデータに重ねる。

③再度 4×4 のデータを復号化しスーパー画素データとする。

かかる処理は何万、何十万というベクトル数を取り扱う場合には極めて時間のかかる処理であり装置全体の処理スピードを著しく低下させる。また、前述の圧縮法は端的に言うとは非可逆的なものであり、何度も圧縮そして復号化を繰り返した場合、当然、画質に劣化が生じる。

一般にグラフィックス・データは線、楕円、矩形等の各種プリミティブパラメータで入力する。例えば円を描く場合、中心座標 (x, y) 、半径 r 、線の種別(実線、点線、一点鎖線etc)、線

400 dpi ($\approx 16 \text{ pel/mm}$)で読み取る画素数であるが(3360×4752 画素)、簡単のため $4k$ ($=4096$) $\times 4k$ であるとする。従ってスーパー画素単位では $1k \times 1k$ の画素数となる。

本実施例における画像データのとりうる圧縮前のアドレス空間は $4k \times 4k$ である。従ってグラフィックスデータも $4k \times 4k$ の空間を自由に描きうるようにするため、とりうる位置 $A(x, y)$ のアドレスは

$$A(x, y) : 0 \leq x \leq 4k, 0 \leq y \leq 4k$$

…①

である。従って x, y 共12ビットのアドレス空間を必要とする。

一方、後述の様に、スーパー画素としてのアドレス空間は $1k \times 1k$ であるためとりうる位置

の太さ、色等の各パラメータを指定して画像メモリ上へ描く位置を計算により求める。この様に各種プリミティブデータに対しては、何を描くかというコマンドとそれを定義するパラメータが付加される。

描画における基本的機能は前述の如く、2点 A, B を結ぶ直線を描くことが基本であり、各種コマンドはこの基本機能を繰り返す事で実現されている。従って前述の2点 A, B を結ぶ線分の描画を高速に出来るか否かがこのシステムのパフォーマンスを決定する重要な要因となる。その意味でこの機能を圧縮メモリ上で行う事は有効ではないと言える。

そこで、本実施例では以下の処理を施して解決する。

本実施例に於て取り扱う画像サイズは $A4$ を

$A'(x', y')$ のアドレスは

$$A(x, y) : 0 \leq x' \leq 1k, 0 \leq y' \leq 1k$$

…②

となる。これは x, y 共に10ビットでよい。本実施例に於ては、この実画素でのアドレス空間 (x, y) と、スーパー画素でのアドレス空間 (x', y') とが

$$x' = x / 4$$

$$y' = y / 4$$

…③

なる関係を満たしており $x \rightarrow x'$ 及び $y \rightarrow y'$ の変換は単に2ビットのシフト演算のみで容易に変換できる。前述した如く、実施例における画像メモリ $40,41,44$ は $1k \times 1k \times 32$ ビットである。従って画像データの $4k \times 4k$ のカラー画像が圧縮されて、この画像メモリ内に格納される。

一方、カラー・モニタ37は圧縮データには前述の様に明度平均及び色度平均をLUT32, 33を介してモニタ37へ出力するため、モニタの解像度は $1k \times 1k$ でよい。従って $4k \times 4k$ のアドレス空間のデータを③式の変換を行い $1k \times 1k$ のアドレス空間に変換し、圧縮操作を行わずそのまま画像メモリ40へ書き込む。

従って第12図のデータフォーマットでそのまま書き込まれる。この時、モニタ37上へ正しく表示させるためLUT32は元のR, G, B信号へ戻す働きをする如くRAMにて構成されている必要がある。

以上の非圧縮での操作は画像メモリ40を単に $1k \times 1k$ の解像度を持つ画像メモリとして用いただけで、このままプリンタ11より出力すれば $1k \times 1k$ の解像度の画像出力しか得られない。

そこで、この実アドレス空間を $1k \times 1k$ の合計16個のブロック単位に切り出し、画像メモリ44にそのうちの1つ、例えば図示のA₀₀ブロックを取り出す。すなわち、画像メモリ44上に先に作成完了したグラフィックデータに基づく線画を描画する(A₀₀ブロックに対しては図示の如く、円の一部分である円弧を描画する)。そして、対応する線画の描画の作成が終了したときには、この画像メモリ44内の画像を圧縮し、対応する画像メモリ40上のブロックB₀₀に格納し、以下、順次、実アドレス空間に対する $1k \times 1k$ のブロックに対して同様の処理を施す。

尚、圧縮を行うために画像メモリ44のサイズはスーパー画素を構成する画素巾の整数倍でなければならない。例えばスーパー画素が 4×4 であれば画像メモリ44の大きさとしては $(4 \times N)$

い。

本来の $4k \times 4k$ の解像度を持った画像出力を行うことをここで再び考える。尚、以下の説明を容易にするため、第14図に第1図の主要部を示す。また、以下の説明ではグラフィック・データ(線画)の作成は全て完了したものとして説明する。

また、この作成済みグラフィックデータにかかると画像は前述した如くモニタ37に表示させてオペレータに確認させるが、そのグラフィックデータそのもの(直線なら、その始点と終点座標等)は例えばCPU1のワークエリア等内に格納させておく。

さて、実際にプリンタ11に出力するときには、第13図(a)に示す様な $4k \times 4k$ の実アドレス空間のサイズを作成しなくてはならない。

$\times (4 \times M)$ となる(但しN, Mは自然数)。このため、実施例における $1k (= 1024) \times 1k$ は問題とならないが、もしスーパー画素が 5×5 であれば 1020×1020 の画像メモリとして用いる様にしないと分割した境界の領域で画像が連続しない。

以上の原理をここでまとめてみると以下の如くなる。

即ち、第13図(a)に示す如く、 $4k \times 4k$ の得るべき画像の領域をA₀₀, A₀₁, A₀₂, ... A₀₃のブロック領域に分けA_{ij}領域に対応するグラフィック・イメージ($1k \times 1k$ 担当分)をソース側の画像メモリ44で作成する。この後、この画像をラスタ走査で読み出して、順次圧縮器115へ出力する。圧縮器115から出力された圧縮データはデスティネーション側の画像メモリ

40の対応する部分に格納される。この格納される領域は第13図(b)に示される $1k \times 1k$ のアドレスを有す圧縮メモリを16分割した $B_{00}, B_{01}, \dots, B_{15}$ のうちの1つの領域 B_{ij} へ圧縮コードとして記憶する。

元々のグラフィックス・データは前述の線分ベクトルの集合体からなる。このベクトル・データを $4k \times 4k$ の画素数をもつ画像に展開して実画像ができ、更に圧縮器を通して最終的に $1k \times 1k$ のスーパー画素数の圧縮画像が得られるわけであるが、実施例の様に、小容量の画像メモリで実現させるために、 $4k \times 4k$ の画像を一旦格納する中間記憶手段が無いため上述の方法をとるわけである。

第15図に上述した変換のフローチャートを示す。尚、このフローチャートに基づくプログラム

こうして、画像メモリ40内に圧縮した高精細の画像が格納されることになる。以下、この画像を保存するときには、SCSIインタフェース21を介してディスク22に書込み、印刷出力するときには、複号化してプリンタ11に出力することで本来の $4k \times 4k$ の画像を得ることが可能となる。

尚、上述した実施例では線画画像のみをその対象としたが、例えばリーダー10より読み込んだ画像(当然圧縮されている)を線画画像とを合成させる様にしても良い。この場合、読み込んだ画像を画像メモリ41に展開し、線画画像を画像メモリ40上で作成し、オペレータに合成表示させて確認させる。そうして、最終的に、線画画像の作成が完了したときには、画像メモリ40上の部分画像を複号して、画像メモリ44に展開し、以

はプログラムメモリ52内に格納されているものである(第1図参照)。

先ず、ステップS1でブロック画像の列方向を特定するための変数 i を“0”に初期化し、続いて、ステップS2で、その行方向を特定するための変数 j を“0”に初期化する。そして、ステップS3に進ん A_{ij} の画像を画像メモリ44で作成し、次いでステップS4で圧縮する。そしてステップS5で圧縮された符号化画像データ(スーパー画素から構成される画像情報)を画像メモリ40内の B_{ij} の位置に格納する。以下、ステップS6で変数 j が“3”であるか否か、そしてステップS8で変数 i が“3”であるか否かを判断し、これらが満足しないうちは、変数 j 或いは i を1インクリメント(ステップS7, S9)して、上述した処理を繰り返す。

下、線画画像を画像メモリ44上で作成して先と同様の処理を施す様にすれば良い。これを実現するためには、第15図のフローチャート中において、ステップS3の直前に、『 A_{ij} の画像を複号し、画像メモリ44に展開する』という処理を挿入させれば良い。

以上、説明した様に本実施例によれば、大量の画像を、それがラスタ構成の自然画像であろうと、ベクトルデータに基づく画像であろうと、小容量の画像メモリを用いて変換及び格納できる様になる。

[発明の効果]

以上、説明した様に本発明によれば、増大する画像データを圧縮することで小容量の画像メモリで処理することを可能とすると共に、ベクトル画像の編集における画像劣化を防ぐことが可能とな

る。

4. 図面の簡単な説明

第1図は実施例における画像処理装置の全体構成図、

第2図は原画を画素単位の構成で示した図、

第3図は実施例における4×4画素領域を単位として圧縮した状態を示す図、

第4図は原画の4×4画素領域とスーパー画素との関係を示した図、

第5図は圧縮後の画像データのフォーマットを示す図、

第6図は実施例における圧縮器の構成を示す図、

第7図はトグルバッファの構成を示す図、

第8図は第6図の検出部の構成を示す図、

第9図はブロック平均算出回路の一例を示す

び21…SCSIインタフェース、22…ディスク、30…CPUバス、31…サブCPUバス、32及び33…ルックアップテーブル、34…合成器、40、41、44…画像メモリ、42…マスクメモリ、50…CPU、51…サブCPU、52…プログラムメモリ、115圧縮器である。

特許出願人 キヤノン株式会社

代理人 弁理士 大塚康徳（他1名）



図、

第10図は第8図に示した回路84の詳細を示す図、

第11図(a)～(d)はベクトル画像の画像メモリへの展開処理を示す図、

第12図は表示時にベクトル画素のフォーマットを示す図、

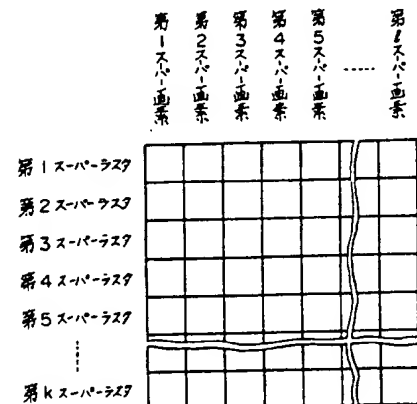
第13図(a)は原画像に対するベクトル部分画像の作成単位を示す図、

第13図(b)はベクトル画像の圧縮後の状態を示した図、

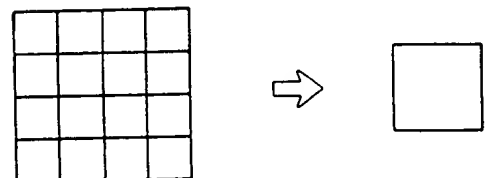
第14図は実施例におけるベクトル画像の処理を行う主要構成図、

第15図は圧縮ベクトル画像の生成に係るフローチャートである。

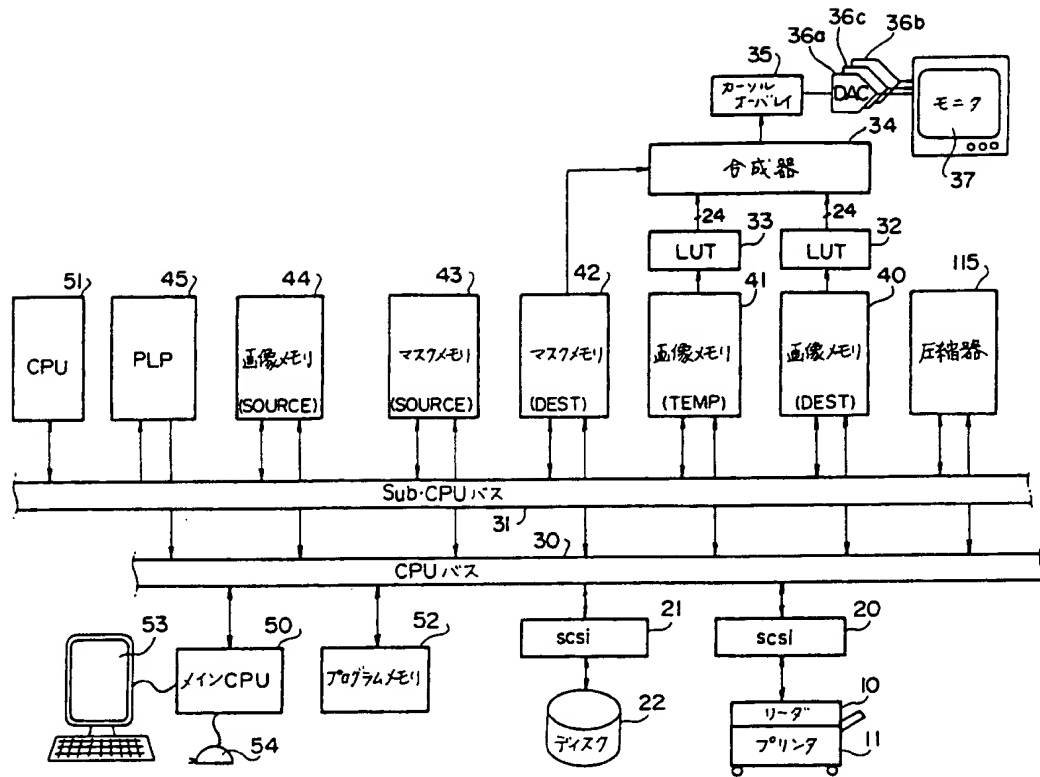
図中、10…リーダー、11…プリンタ、20及



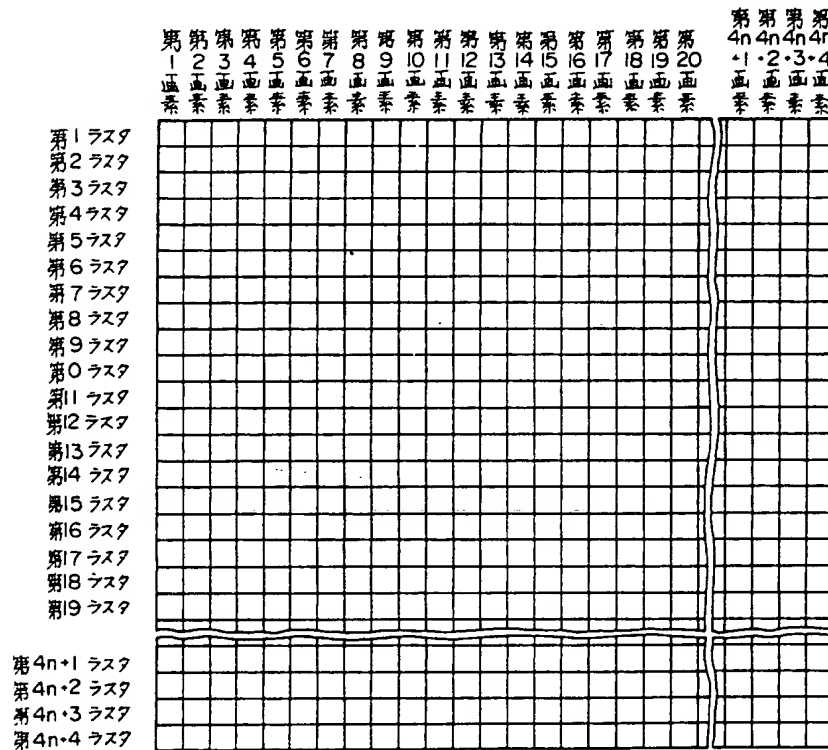
第3図



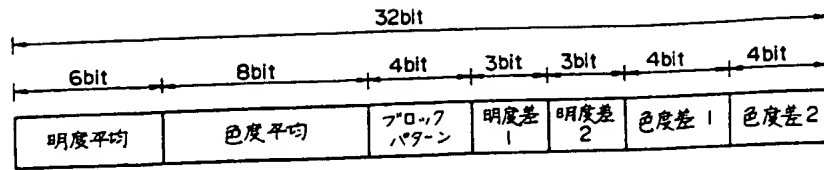
第4図



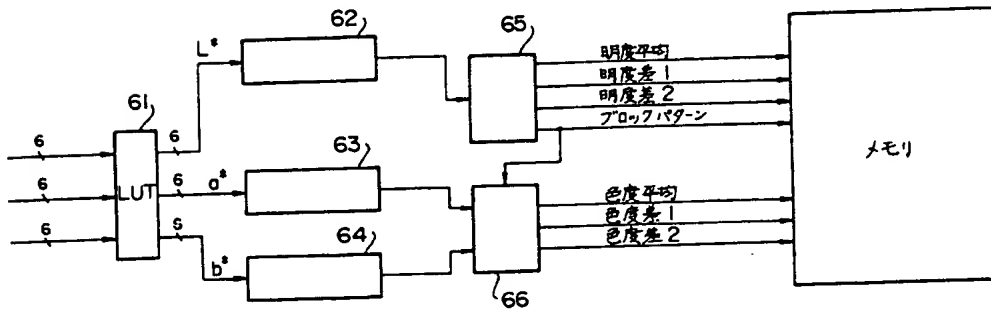
第 1 図



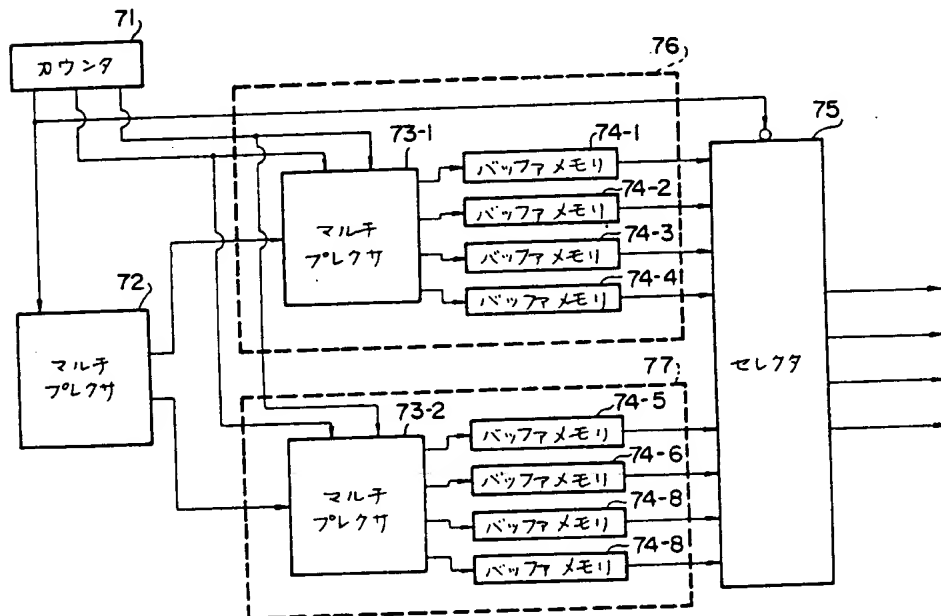
第 2 図



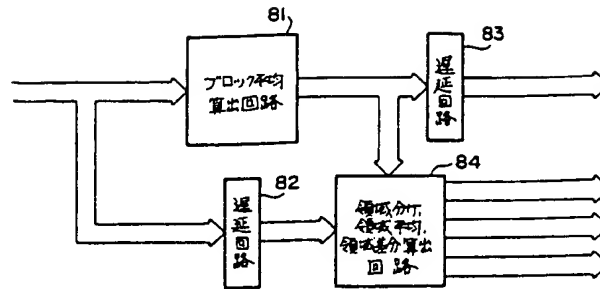
第 5 図



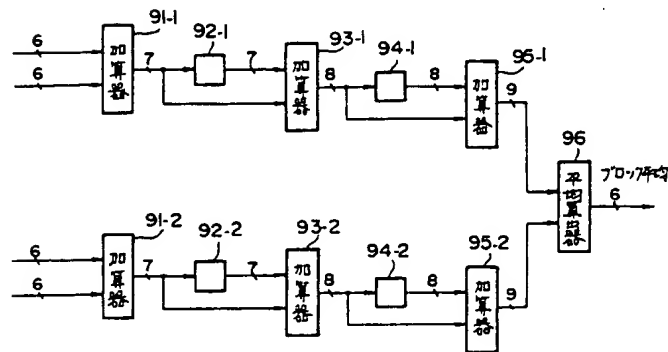
第 6 図



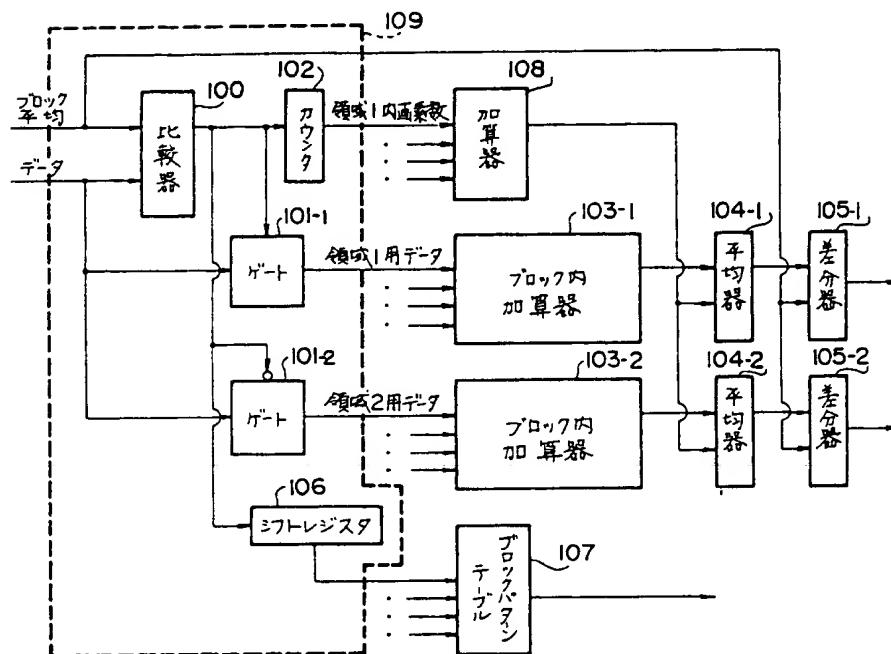
第 7 図



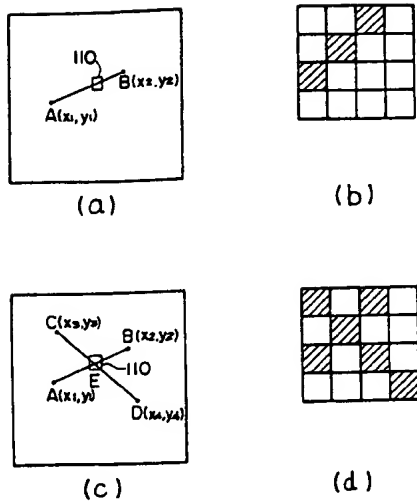
第 8 図



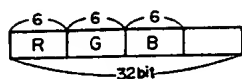
第 9 図



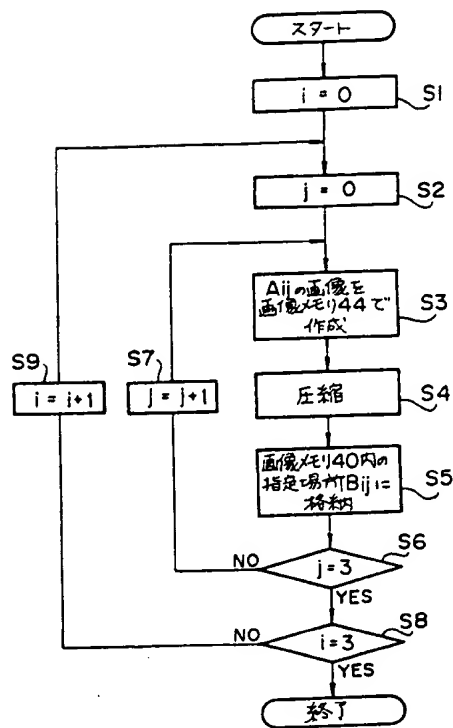
第 10 図



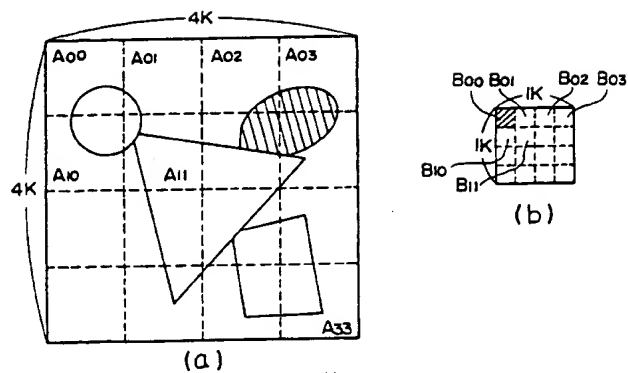
第11図



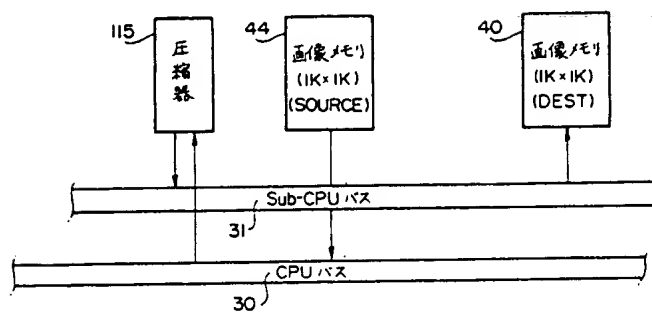
第12図



第15図



第13図



第14図

(19) Japan Patent Office (JP)

(12) Publication of Unexamined Patent Application (A)

(11) Japanese Patent Laid-Open Number: Hei 1-311379 (P1989-311379A)

(43) Laid-Open Date: Heisei 01-12-15 (Dec. 15, 1989)

(51) Int.Cl.⁴ Identification Code Int. Reference No.

G06F 15/66 330 H-8419-5B

15/72 350 8125-5B

H04N 1/21 8839-5C

1/411 7060-5C

Request for examination: No request to be done

Number of Invention: 1 (14 pages in total)

(54) Title of the Invention: Image recording device

(21) Application Number: Sho 63-141825

(22) Filed Date: Sho 63-06-10 (June 10, 1988)

(72) Inventor: Naoto Kawamura

Canon Inc.

3-30-2 Shimomaruko, Ohta, Tokyo

(71) Applicant: Canon Inc.

3-30-2 Shimomaruko, Ohta, Tokyo

(74) Agent: Patent Attorney; Yasunori Otsuka

SPECIFICATION

1. Title of the Invention: Image processing apparatus

2. Claims

An image processing apparatus, comprising:

an image memory having a capacity for at least a partial image of a vector image;

a development means for developing a vector image corresponding to the partial image within the image memory;

a compression means for compressing the partial vector image developed by the development means; and

a storage means for storing the compressed partial vector image so as to be associated with the entire image ,

wherein the image processing apparatus is characterized in allowing all compressed partial vector images to the vector image corresponding to be stored in the storage means.

3. Detailed Description of the Invention

[Industrial Field]

The present invention relates to an image processing apparatus. In particular, the present invention relates to an image processing apparatus for processing an image with an image memory having a small capacity.

[Prior Art]

In recent years, with the development of image processing techniques, a displayed image or an outputted image print is required to have a higher definition. Thus, it is presumed that the capacity of such an image memory will be extremely large when an image is displayed on a display screen or an image is outputted into a print.

For example, when an attempt is made to output an image of 400 dpi into A4 size, the number of pixels composing the image is as much as about 16,000,000. Specifically, when an attempt is made to provide image data by 8 bit/image, an image memory having the capacity of about 16 megabytes is required to provide an image for one page of A4 size and an image memory having the capacity of about 48 megabytes is required to provide a color image of 8 bits of R, G, and B, respectively. This requires a larger apparatus and more cost as well as longer time for the communication.

[Problem to be Solved by the Invention]

Thus, it is considered to compress and store an original image by compressing 3×3 pixels to one display pixel, for example. However, when the memory having such a manner of storage tries to develop a vector image, the problems as shown below are caused.

In other words, a processing in which compressed data is decoded to write predetermined dots to encode (compress) the data again must be performed with all dots of the one vector image, thereby causing the apparatus to have a remarkably

lowered processing rate.

Furthermore, the above-described processing must be repeated when individual vector image is developed, which causes a gradual deterioration of the image itself.

The present invention was made in view of such problems. It is an object of the present invention to provide an image processing apparatus which allows, by compressing such increasing image data, an image memory having a small capacity to process an image and which can prevent a vector image from being deteriorated in editing the image while reducing the image deterioration.

[Means for Solving the Problem]

In order to solve the problems, the present invention includes the structure as shown below.

Specifically, the present invention includes: an image memory having a capacity for at least a partial image of a vector image; a development means for developing a vector image corresponding to the partial image within the image memory; a compression means for compressing the partial vector image developed by the development means; and a storage means for storing the compressed partial vector image so as to be associated with the entire image.

[Operation]

In such a structure of the present invention, on an image memory having the capacity for a partial image, a vector image corresponding to the partial image is developed. Then, the developed vector image is compressed to allow the compressed partial image to be associated with an entire image for storage.

[Embodiment]

Hereinafter, embodiments of the present invention will be described.

<Description of the basic structure (Figure 1)>

First, the structure of an image editing apparatus in the present embodiment is shown in Figure 1. The following section describes the outline of the operation.

Image data read out by a reader 10 is compressed by a compressor 11, and then is loaded to an image memory 40 or 41 via a general-purpose SCSI interface 20. The contents of the image memories 40 and 41 are subjected to a color conversion

processing by 1 okup tables (hereinafter simply referred to as LUT) 32 and 33, and then are recovered to be R, G, and B analog signals by D/A converters 36a, 36b, and 36c, thereby being displayed on a monitor apparatus 37.

Then, in order to enhance the editing function, the image memory in the embodiment has three image memories, a for-source image memory 44, a for-intermediate-buffer image memory 41, and a for-destination image memory 40, each of which has a capacity for one page. Each of the memories is selectively connected to two buses (a CPU bus 30 and sub-CPU bus 31). Such selection is performed by an order from a main CPU 50 and via a control BUS existing in the bus 30. Specifically, each of the image memories 40, 41, and 44 is always connected, by the order from the main CPU 50, to the CPU bus 30 and the sub-CPU bus 31, respectively. On the monitor apparatus 37, two image memories 40 and 41 are synthesized by the information in a mask memory 42, thereby being displayed. At this moment, the information in the mask memory 42 is read out in the synchronization with the image memories 40 and 41 at the video rate. Whether the image memory 40 or the image memory 41 is displayed is determined depending on the contents in the mask memory 42. Thus, the output from the mask memory 42 is used to allow a synthesizer 34 to select the read data of the two image memories, thereby allowing two images to be synthesized on the monitor 37.

<Description of compression of image data (Figure 2 to Figure 10)>

Next, the compression of image data will be described.

Figure 2 is a diagram illustrating an original image on a pixel basis. Figure 3 is a diagram illustrating a status in which a region of 4×4 pixels of an original image (hereinafter this pixel region is referred to as a super pixel) is used as a unit to compress the region and the compressed data is structured as super pixel data on a super pixel basis. Figure 4 is a diagram illustrating the correspondence between a region of 4×4 pixels of an original image and a super pixel.

In the present embodiment, a high definition image is stored in each memory in a compressed manner and the compressed data is allowed to be displayed on the display while allowing the editing operation to be performed, thereby reducing the costs for memories and the display apparatus.

Figure 5 illustrates an example of compressed data. The compressed data is structured to have a data length (each super pixel) of 32 bits. For a region consisting of 4x4 pixels of an original image of this data, an average brightness value is 6 bits; an average chroma value is 8 bits; block pattern information is 4 bits; brightness differences 1 and 2 are 3 bits, respectively; and chroma differences 1 and 2 are 4 bits, respectively. Each pixel of the original image is data having a total of 18 bits of color information consisting of pieces of 6 bits color information of R, G, and B. Thus, an image of 4x4 pixels totals up to total of 288 bits ($=18 \times 16$) and is compressed to be 1/9.

Figure 6 is a block structure diagram of a compressor 115 to prepare compressed data to store the compressed data in a memory from each 6 bits of R, G, and B data of an original image.

In Figure 6, "61" refers to a lookup table for inputting and outputting 18 bits; "62" to "64" refer a toggle buffer consisting of two pairs of four line buffers, respectively; and "65" refers to a detection section for using brightness data to divide an brightness average and 4x4 pixels into the pixel regions having larger values than those of the brightness average and the pixel regions having smaller values than those of the brightness average (binarization), thereby outputting the boundary information of the region as block pattern information. By the detection section 65, the difference information of the brightness average of each of the pixel regions separated by the boundary and the brightness average of the above-described 4x4 pixel-region is outputted as a brightness difference 1 and brightness difference 2. Here, the brightness difference 1 is the difference information of a region having a value larger than an average while the brightness difference 2 means the difference information of a region having a value smaller than an average. The brightness data is calculated by a uniform color space $L^*a^*b^*$ and chroma information which will be described later is calculated using a^*b^* . The conversion from R, G, and B data of the original image to $L^*a^*b^*$ is performed using LUT 61.

Hereinafter, a method for preparing compressed data will be described in further details.

Figure 7 is a diagram for explaining in detail the structure of the toggle buffers 62 to 64.

In Figure 7, "74-1" to "74-4" and "74-5" to "74-8" refer buffer memories and are pairs of buffer memories (the following description will refer the buffer memories 74-1 to 74-8 as a buffer group 77).

Data input to each buffer memory is performed by the switching between multiplexers 73-1 and 73-2. The multiplexer 72 sends data by determining a buffer group 76 as a destination of the inputted image data or by selecting the output of a buffer group 77 and outputs the outputs from the four buffers in parallel. A counter 71 is used for controlling the multiplexers 72, 73-1, and 73-2 and a selector 75. The counter 71 counts raster synchronization signals to switch, for every four rasters, the multiplexer 72 and the selector 75, respectively and also switches, for every one raster, the destination (buffer memory) of the outputs from the multiplexer 73-1 (and multiplexer 73-2).

Figure 8 is a diagram for explaining the detection section 65 of Figure 6 for outputting the brightness information from the data from the four lines of L^* . In Figure 8, "81" refers to a block average calculation circuit for calculating the average of L^* within a block; and "82" refers to a delay circuit for providing a delay by a time during which the block average calculation circuit 81 needs to calculate the average L^* within a block. Reference numeral "84" refers to a circuit for a calculation, based on L^* data of each pixel in a block and the output from the block average calculation circuit 81, the difference value of a region separation within a block, the region average of the each region, and a block average. These provide the brightness differences 1 and 2 as shown in Figure 5.

Figure 9 is a block diagram illustrating an example of the block average calculation circuit 81.

Here, data of four rasters are inputted in parallel to input the data of 16 pixels (4×4 pixels).

First, adders 91-1 and 91-2 respectively add, on a two pixels basis, data of 4×4 pixels inputted in parallel. Buffers 92-1 and 92-2 respectively execute a temporary storage of the output from the adders 91-1 and 91-2, respectively. Next, an adder 93-2 adds a series of two outputs of the outputs from each of the adder 91-1 and 91-2. Buffers 94-1 and 94-2 hold temporarily the output from the adders 93-1 and 93-2.

Adders 95-1 and 95-2 add a series of two outputs from the adders 93-1 and 93-2. Finally, an average calculator 96 is used to add the outputs from the adders 95-1 and 95-2 to calculate the average. Here, the adders 93-1 and 93-2 operate with a cycle twice as that of the addition by the adders 91-1 and 91-2. The adders 95-1 and 95-2 and the average calculator 96 also operate with a cycle twice as those of the addition by the adders 91-1 and 91-2. Thus, when the adders 91-1 and 91-2 perform four additions, the adders 93-1 and 93-2 operate two times. A value on the bus for connecting the adders 95-1 and 95-2, the average calculator, and the buffer represent the number of bits of the bus.

Figure 10 illustrates the details of the circuit 84 shown in Figure 8 for calculating a region separation, a region average, and a region difference which calculates a region separation, a region average, and a region difference.

In Figure 10, "100" refers to a comparator which inputs the data for each pixel within the block from the delay circuit 82 shown in Figure 8 and inputs the block average from the block average calculation circuit 81 to judge whether each pixel is in a region having a value larger than the block average or in a region having a value equal to/less than the block average. One block is composed of four rasters and each raster is composed of four pixels. Thus, a description in the following section will be made on the assumption that each raster has one comparator 100. In other words, the comparator 100 of Figure 10 and counters or the like at the periphery provide a processing to one raster.

Incidentally, the output from each comparator 100 is outputted as a switching signal for two gates 101-1 and 101-2, respectively. The same pixel data is inputted to both of the gates 101-1 and 101-2. Specifically, the output from the comparator 100 allows either of the gates 101-1 and 101-2 to output the inputted data as it is and the other to output "0". A counter 102 counts the result of the comparator 100 to count how many pixels among four pixels are in the region 1 (region having a value larger than the block average). A shift register 106 shifts the output from the comparator (binarization) to output the shift statuses in parallel. This allows the block patterns of the region 1 and region 2 within a block (region having a value equal to or less than the block average) to be represented. The region 109 surrounded by the broken lines as

described above exists in each raster.

Meanwhile, the adder 108 adds all of the numbers of pixels within the region 1 of each raster to output to averaging devices 104-1 and 104-2 the number of pixels in the block. The adders 103-1 and 103-2 respectively calculate the sum of the values of each pixel in the region 1 and each pixel in the region 2 within a block (a value equal to or less than the block average). The averaging device 104-2 inputs the number of pixels to which the regions 1 and 2 belong to output the average value. In addition, the adders 103-1 and 103-2 can be provided by a circuit similar to that shown in Figure 9. Thereafter, the output from the adder of "96" is outputted without omitting bits. Difference devices 105-1 and 105-2 are used for outputting the difference between the average of the regions 1 and 2 and the block average, respectively. A block pattern table 107 is used for inputting a pattern of 4 bits for each raster and for outputting a pattern of a total of 16 bits as a pattern code. The averaging devices 104-1 and 104-2, the difference devices 105-1 and 105-2, and the block pattern table 107 can be easily realized by the LUT (lookup table) of a ROM.

For chroma, the structure as described above also can be used in a similar way. However, a signal for separating the region 1 and the region 2 is provided by the data for brightness and the data is provided by the chroma data a' and b' . The data a' and b' are handled individually, and the data a' and b' are collectively handled as chroma data.

The compressed data thus obtained is compressed as described above and is used as display data to read a A4 size paper, for example, with the image density of 16 pixels/mm. The result is that the conventionally-provided data amount of 4752×3360 pixels can be substituted by 1188×840 super pixels, thereby providing a compression ratio reduced to $1/9$.

The above-described compression method allows the pixel arrangement of the image data of Figure 2 to be converted to the super pixel arrangement of Figure 3 to have a smaller capacity. For example, data for one A4 page corresponds to about 4M (mega)bytes ($\approx 1188 \times 840 \times 32$ bits). This can be easily realized by an IC memory. Each of the image memories 40, 41, and 43 in Figure 1 is composed of this capacity, respectively.

When an compressed image stored in the image memory 40 by the above-described processing (which is assumed as an image A) and an compressed image stored in the image memory 41 (which is assumed as an image B) are synthesized and displayed, the image A and the image B are switched by the synthesizer 34 in accordance with the data in the mask memory 42, thereby being displayed on the monitor apparatus 37. At this time, the image A and the image B have data lengths of 32 bits as described above (Figure 5). However, when the image A and the image B are displayed, only the brightness average and the chroma average are outputted and other block patterns or the like are decrypted when the images are printed.

Here, when such an image is printed, the block pattern within the data length of 32 bits (in which a code is stored) is once decrypted to be 4x4 of binarized blocks and the brightness value to the pixel represented as "1" is to be a value obtained by allowing the brightness average to be added with the brightness difference 1 and the chroma value is to be a value obtained by allowing the chroma average to be added with the chroma difference 1, thereby providing the decryption.

On the other hand, the pixel represented as "0" is decrypted by allowing the brightness value to be a value obtained by deducting the brightness difference 2 from the brightness average and by allowing the chroma difference to be a value obtained by deducting the chroma difference 2 from the chroma average.

These processing are easily achieved using the lookup table.

<Description of vector image (Figure 11 to Figure 15)>

As described above, the present embodiment described an image to be processed as a natural image (image from the reader 10). Then, two-dimensional image data is compressed and encoded by a raster scanning, thereby allowing the image data having an increasing capacity to have a smaller capacity.

Incidentally, there is a disadvantage in which, when image data is CAD and/or Graphics composed of graphics data and vector font called an outline font or the like (e.g., line image), the conversion by compression and encoding is more complicated.

With reference to Figures 11(a) to 11(d), this defect will be briefly described.

Now, as shown in Figure 11(a), it is assumed that a segment vector AB

connecting two points, $A(x_1, y_1)$ and $B(x_2, y_2)$, is drawn on the image memory.

When the image memory is an uncompressed and normal one, the straight line connecting the two points A and B is calculated by the following equation:

$$y = [y_2 - y_1 / x_2 - x_1](x_2 - x_1) + y_1 \quad (\text{where } x_1 \leq x \leq x_2)$$

Then, the address of (x, y) represented as integers may be written with the information for the luminance of the line (or density) or color information.

On the other hand, a case will be described in which the image memory is the one on a super pixel basis according to the above-described compression method. In this case, as shown in Figure 11(b), image drawing information within a super pixel consisting of 4×4 pixels must be sent to the compressor 115 to be encoded, and then must be sent to the image memory. At this time, as shown in Figure 11(c), when a segment vector CD is overwritten on the already-drawn segment vector AB, the base code data at the node E (110) must be used to convert the code data representing the superimposed segments (Figure 11(d)). Specifically, the data for the node E (110) is already encoded when the vector AB is drawn. When another vector CD is overwritten thereon, the following processing will be conducted.

- (1) The super pixel data including the node E is decrypted to be recovered to be original 4×4 data.
- (2) Data to be overwritten is superposed on this 4×4 data.
- (3) The 4×4 data is decrypted again to be super pixel data.

Such processing is very time-consuming when handling tens of thousands or hundreds of thousands of vectors, which remarkably lowers the processing speed of the entire apparatus. The above-described compression method is also the irreversible when simply stated. Thus, repeated compression and decryption using this method brings deterioration in image quality.

Generally, graphics data is inputted by various primitive parameters such as lines, ellipses, or rectangles. When a circle is drawn for example, various parameters, e.g., center coordinates (x, y) , radius r , the line type (solid line, dotted line, dashed line etc), the line thickness, color, are specified to calculate the position on the image

memory in which the circle is drawn. In this way, various primitive data is added with a command representing what is to be drawn and the parameter defining it.

The basic function in the image drawing is as described above in which a straight line connecting two points A and B is drawn and various commands are realized by repeating this basic function. Thus, whether the above-described image drawing of a segment connecting two points A and B can be performed with a high speed or not is an important factor determining the performance of this system. In this sense, it can be said that performing this function on a compressed memory is not effective.

Thus, the present embodiment performs the following processings to solve the disadvantage.

For the size of an image to be handled by the present embodiment, the number of pixels of an image for A4 size is read by 400 dpi (≈ 16 pel/mm) (3360x4752 pixels). For simplicity, it is assumed to be $4k(=4096) \times 4k$. Thus, representing the image on a super pixel basis results in the number of pixels of $1k \times 1k$.

The address space which may be taken by the image data before compression in the present embodiment is $4k \times 4k$. Thus, graphic data also may be used to freely draw the space of $4k \times 4k$ and thus the address of a position $A(x, y)$ which may be taken is:

$$A(x, y): 0 \leq x \leq 4k, 0 \leq y \leq 4k \quad \text{formula 1}$$

Thus, both of "x" and "y" require the address space of 12 bits.

On the other hand, as described later, the address space as a super pixel is $1k \times 1k$, and thus the address of a position $A'(x', y')$ which may be taken is:

$$A(x, y): 0 \leq x' \leq 1k, 0 \leq y' \leq 1k \quad \text{formula 2}$$

This may be 10 bits for both of "x" and "y". In the present embodiment, this address space (x, y) with actual pixels and the address space (x, y) with super pixels satisfy the following relation:

$$x' = x/4$$

$$y' = y/4$$

formula 3

Thus, the conversion of "x to x'" and "y to y'" can be easily performed by the shift calculation with 2 bits. As described above, the image memories 40, 41, and 44 in the embodiment are composed of $1k \times 1k \times 32$ bits. Thus, $4k \times 4k$ of a color image of image data is compressed and is stored in this image memory.

On the other hand, for compressed data, the color monitor 37 outputs the brightness average and the chroma average to monitor 37 via the LUT 32 and 33 as described above, and thus the monitor may have a resolution of $1k \times 1k$. Thus, data of $4k \times 4k$ of an address space is subjected to the conversion according to the formula 3 to be converted to $1k \times 1k$ of an address space, thereby being written to the image memory 40 as it is without performing a compression operation.

Accordingly, the data is written as it is with the data format of Figure 12. At this time, in order to provide a correct display on the monitor 37, the LUT 32 is required to be structured by RAM's so as to work to provide the recovery to original R, G, and B signals.

The above operation for uncompressed data merely uses the image memory 40 as an image memory having the resolution of $1k \times 1k$. Thus, when the data is outputted from the printer 11 as it is, the image output only with $1k \times 1k$ resolution is obtained.

Now, it is reconsidered to perform the image output with the original $4k \times 4k$ resolution. Note that, in order to provide the following description in a simpler manner, the main sections of Figure 1 are shown in Figure 14. Moreover, the following description is made on the assumption that all of the preparation of the graphic data (line image) has been completed.

The image according to this already-prepared graphic data is displayed on the monitor 37 as described above so that the image is checked by an operator. However, the graphic data itself (the coordinates of the start and end points in a case of a straight line) is stored in the work area or the like of the CPU 1, for example.

Meanwhile, when the data is actually outputted to the printer 11, then the size of an actual address space of $4k \times 4k$ as shown in Figure 13 must be prepared.

Therefore, this actual address space is cut to a total of 16 blocks of $1k \times 1k$ and one of them (e.g., A_{00} block shown in the drawing) is taken out. Specifically, the line image based on the previously-completed graphic data is drawn on the image memory 44 (a circular arc, a part of a circle, is drawn to A_{00} block as shown in the drawing). Then, when the preparation of the drawing of the corresponding line image is completed, the image in this image memory 44 is compressed and is stored in the block B_{00} on the corresponding image memory 40. Thereafter, $1k \times 1k$ blocks to actual address spaces are consecutively subjected to the same processing.

It should be noted that, in order to provide the compression, the size of the image memory 44 must be an integral multiple of the pixel width configuring a super pixel. When a super pixel is 4×4 for example, the size of the image memory 44 is $(4 \times N) \times (4 \times M)$ (N and M are natural numbers). Thus, $1k (= 1024) \times 1k$ in the embodiment is not a problem. However, if a super pixel is 5×5 , the use as an image memory of 1020×1020 is required in order to provide successive images at separated regions at a boundary.

The following section describes the summary of the above principles.

Specifically, as shown in Figure 13(a), a region of an image to be obtained for $4k \times 4k$ is separated to block regions of $A_{00}, A_{01}, A_{02}, \dots, A_{33}$ to prepare the graphic image corresponding to the A_{ij} region (which corresponds to $1k \times 1k$) by a source-side image memory 44. Thereafter, this image is read out by a raster scanning and is consecutively outputted to the compressor 115. The compressed data outputted from the compressor 115 is stored in the corresponding section of a destination-side image memory 40. This region in which the data is stored stores the data by storing the data as a compressed code in one region B_{ij} among $B_{00}, B_{01}, B_{02}, \dots, B_{33}$ which are obtained by dividing the compressed memory having an address of $1k \times 1k$ shown in Figure 13(b) into sixteen regions.

The original graphics data is formed of the collection of the above-described segment vectors. This vector data is developed to be an image having the number of pixels of $4k \times 4k$ to provide an actual image, which is further sent to the compressor to finally provide a compressed image having the number of super pixels of $1k \times 1k$. However, as shown in the embodiment, since the processing with an image memory

having a small capacity is realized, there is no intermediate storage means for once storing an image of $4k \times 4k$. Thus, the above-described method is used.

Figure 15 illustrates a flow chart of the above-described conversion. The program based on this flow chart is stored in a program memory 52 (see Figure 1).

First, Step S1 allows a variable "i" for specifying the column direction of a block image to be initialized to "0" and then Step S2 allows a variable "j" for specifying the row direction to be initialized to "0". Then, the processing proceeds to Step S3 in which an image of A_{ij} is prepared by the image memory 44 and then the image is compressed by Step S4. Thereafter, Step S5 allows the compressed encoded image data (image information composed of super pixels) to be stored in the position of B_{ij} in the image memory 40. Thereafter, Step S6 judges whether the variable "j" is "3" or not and Step S8 judges whether the variable "j" is "3" or not. Until these are satisfied, the variable "j" or "i" is incremented by one (Steps S7 and S9) to repeat the above-described processing.

In this way, the image memory 40 has therein the compressed high definition image. Thereafter, when this image is stored, the data is written to a disc 22 via the SCSI interface 21. When the image is outputted as a print, the data is decoded to be outputted to the printer 11. As a result, the original image of $4k \times 4k$ can be obtained.

The above-described embodiment described only an image of a line image as an image to be processed. However, the image which is read by the reader 10 (which is compressed as a matter of course), for example, may be synthesized with an image of a line image. In this case, the read image is developed to the image memory 41 and the image of a line image is prepared on the image memory 40, thereby allowing the operator to synthesize and display the image for confirmation. Then, when the preparation of the image of a line image is finally completed, a partial image on the image memory 40 is decoded to be developed to the image memory 44. Thereafter, an image of a line image may be prepared on the image memory 44 to be subjected to the processings as described above. In order to realize this, in the flow chart of Figure 15, the section immediately prior to Step S3 may be inserted with a processing of the image of " A_{ij} " is decoded to be developed to the image memory 44.

As described above, according to the present embodiment, a large amount of

image can be converted and stored using an image memory having a small capacity even if the image is a natural image having a raster structure or an image based on vector data.

[Effects of the Invention]

As described above, according to the present embodiment, increasing image data is compressed to allow the data to be processed by an image memory having a small capacity. At the same time, image deterioration in editing vector image also can be prevented.

4. Brief Description of the Drawings

Figure 1 is a diagram illustrating the entire structure of an image processing apparatus in the embodiment.

Figure 2 is a diagram illustrating an original image with the structure on a pixel basis.

Figure 3 is a diagram illustrating the status in which compression is executed on a 4x4 pixel region basis in the embodiment.

Figure 4 is a diagram illustrating the relation between 4x4 pixel region of an original image and a super pixel.

Figure 5 is a diagram illustrating a format of the compressed image data.

Figure 6 is a diagram illustrating the structure of the compressor in the embodiment.

Figure 7 is a diagram illustrating the structure of the toggle buffer.

Figure 8 is a diagram illustrating the structure of the detection section of Figure 6.

Figure 9 is a diagram illustrating an example of the block average calculation circuit.

Figure 10 is a diagram illustrating the details of the circuit 84 shown in Figure 8.

Figures 11(a) to 11(d) are diagrams illustrating a processing in which a vector image is developed to an image memory.

Figure 12 is a diagram illustrating a format of the vector pixel during the display.

Figure 13(a) is a diagram illustrating a preparation unit of a vector partial image to an original image.

Figure 13(b) is a diagram illustrating the status after the compression of the compressed vector image.

Figure 14 is a diagram showing the main structure for processing a vector image in the embodiment.

Figure 15 is a flow chart of the generation of a compressed vector image.

Reference numerals in the drawings are described below:

10	Reader
11	Printer
20 and 21	SCSI interface
22	Disc
30	CPU bus
31	Sub-CPU bus ,
32 and 33	Look-up table
34	Synthesizer
40, 41, and 44	Image memory
42	Mask memory
50	CPU
51	Sub-CPU
52	Program memory
115	Compressor

Figure 1

10	Reader
11	Printer
22	Disc
30	CPU bus
31	Sub-CPU bus
34	Synthesizer
35	Cursor overlay
37	Monitor
40	Image memory
41	Image memory
42	Mask memory
43	Mask memory
44	Image memory
50	Main CPU
52	Program memory
115	Compressor

Figure 2

1 st pixel
2 nd pixel
3 rd pixel
4 th pixel
5 th pixel
6 th pixel
7 th pixel
8 th pixel
9 th pixel
10 th pixel
11 th pixel
12 th pixel

13th pixel
14th pixel
15th pixel
16th pixel
17th pixel
18th pixel
19th pixel
20th pixel
4n+1-th pixel
4n+2-th pixel
4n+3-th pixel
4n+4-th pixel
1st raster
2nd raster
3rd raster
4th raster
5th raster
6th raster
7th raster
8th raster
9th raster
10th raster
11th raster
12th raster
13th raster
14th raster
15th raster
16th raster
17th raster
18th raster
19th raster

4n+1-th raster

4n+2-th raster

4n+3-th raster

4n+4-th raster

Figure 3

1st super pixel

2nd super pixel

3rd super pixel

4th super pixel

5th super pixel

"l"-th super pixel

1st super raster

2nd super raster

3rd super raster

4th super raster

5th super raster

"k"-th super raster

Figure 5

Brightness average

Chroma average

Block pattern

Brightness difference 1

Brightness difference 2

Chroma difference 1

Chroma difference 2

Figure 6

Brightness average

Brightness difference 1

Brightness difference 2
Block pattern
Chroma average
Chroma difference 1
Chroma difference 2
Memory

Figure 7

71 Counter
72 Multiplexer
73-1 Multiplexer
73-2 Multiplexer
74-1 Buffer memory
74-2 Buffer memory
74-3 Buffer memory
74-4 Buffer memory
74-5 Buffer memory
74-6 Buffer memory
74-7 Buffer memory
74-8 Buffer memory
75 Selector

Figure 8

81 Block average calculation circuit
82 Delay circuit
83 Delay circuit
84 Circuit for calculating difference of region separation, region average, and block average

Figure 9

91-1 Adder

93-1 Adder
95-1 Adder
91-2 Adder
93-3 Adder
95-4 Adder
96 Average calculator
Block average

Figure 10

Block average

Data

100 Comparator

101-1 Gate

101-2 Gate

Data for region 1

Data for region 2

102 Counter

Number of pixels within region 1

103-1 Within-block adder

103-2 Within-block adder

104-1 Averaging device

104-2 Averaging device

105-1 Difference device

105-2 Difference device

106 Shift register

107 Block pattern table

108 Adder

Figure 14

30 CPU bus

31 Sub-CPU bus

40 Image memory
44 Image memory
115 Compressor

Figure 15

Start

S3 Image of A_{ij} is prepared by image memory 44.

S4 Compression

S5 Storage in designated position B_{ij} in image memory 40.

Completion

End

る。

4. 図面の簡単な説明

第1図は実施例における画像処理装置の全体構成図、

第2図は原画を画素単位の構成で示した図、

第3図は実施例における4×4画素領域を単位として圧縮した状態を示す図、

第4図は原画の4×4画素領域とスーパー画素との関係を示した図、

第5図は圧縮後の画像データのフォーマットを示す図、

第6図は実施例における圧縮器の構成を示す図、

第7図はトグルバッファの構成を示す図、

第8図は第8図の検出部の構成を示す図、

第9図はブロック平均算出回路の一例を示す

図21…SCSIインタフェース、22…ディスク、30…CPUバス、31…サブCPUバス、32及び33…ルックアップテーブル、34…合成部、40、41、44…画像メモリ、42…マスクメモリ、50…CPU、51…サブCPU、52…プログラムメモリ、115圧縮器である。

特許出願人 キヤノン株式会社
代理人 弁理士 大塚廣徳 (他1名)



図、

第10図は第8図に示した回路84の詳細を示す図、

第11図(a)～(d)はベクトル画像の画像メモリへの展開処理を示す図、

第12図は表示時にベクトル画像のフォーマットを示す図、

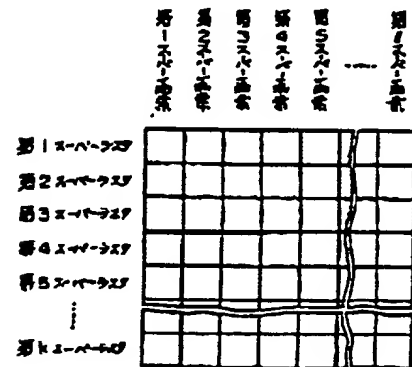
第13図(a)は原画像に対するベクトル部分画像の作成単位を示す図、

第13図(b)はベクトル画像の圧縮後の状態を示した図、

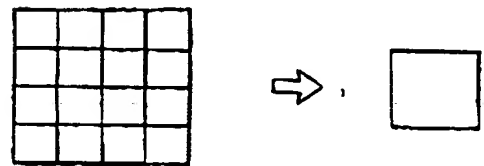
第14図は実施例におけるベクトル画像の処理を行う主要構成図、

第15図は圧縮ベクトル画像の生成に係るフローチャートである。

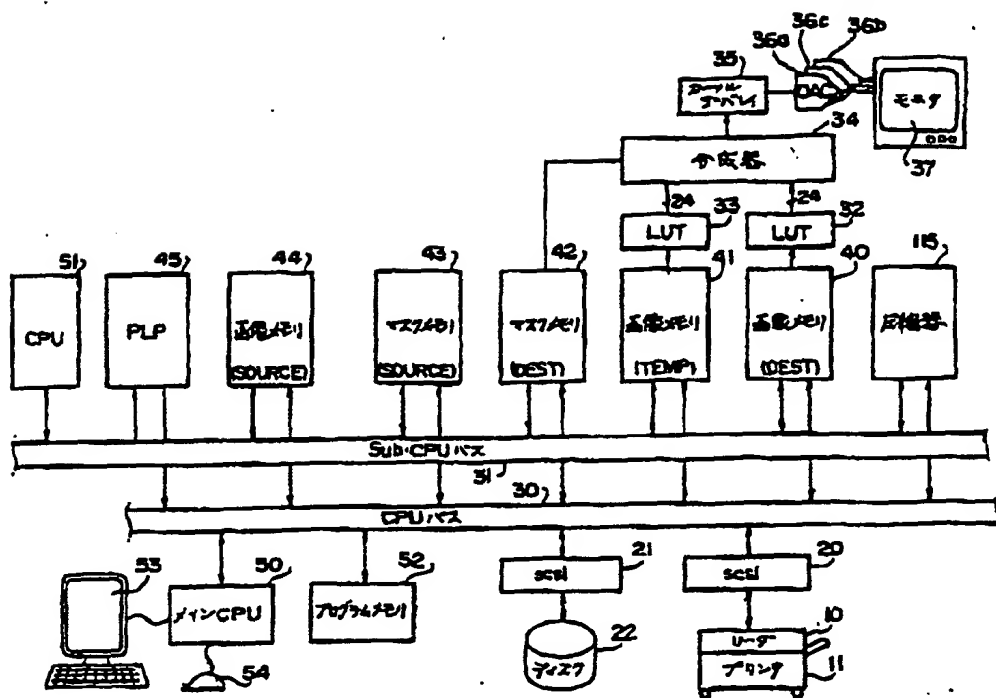
図中、10…リーダー、11…プリンタ、20及



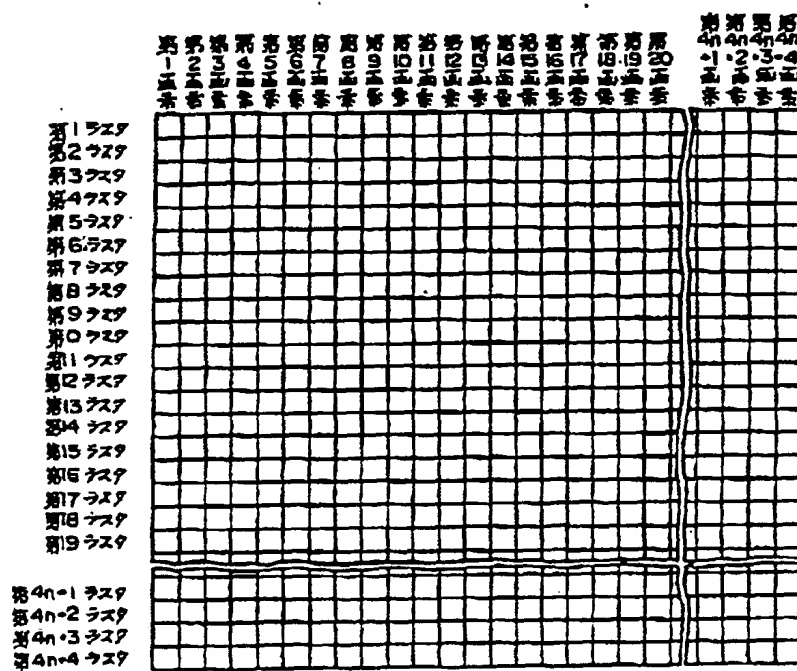
第3図



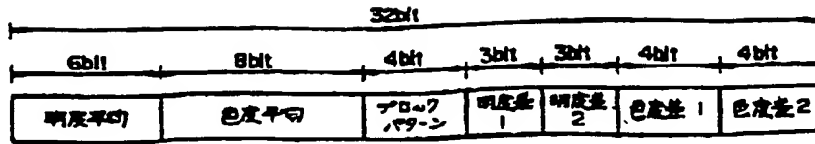
第4図



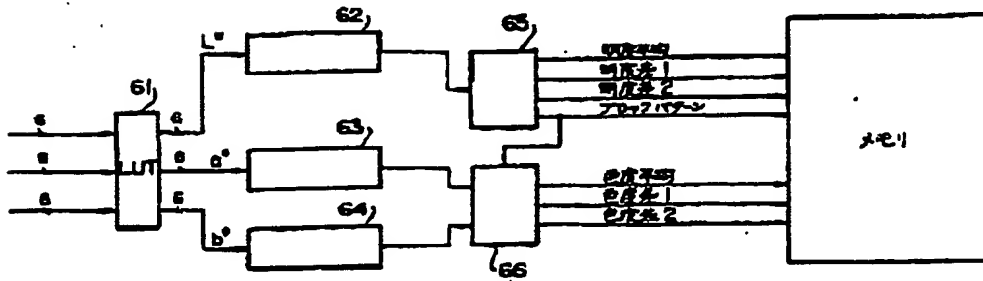
第 1 図



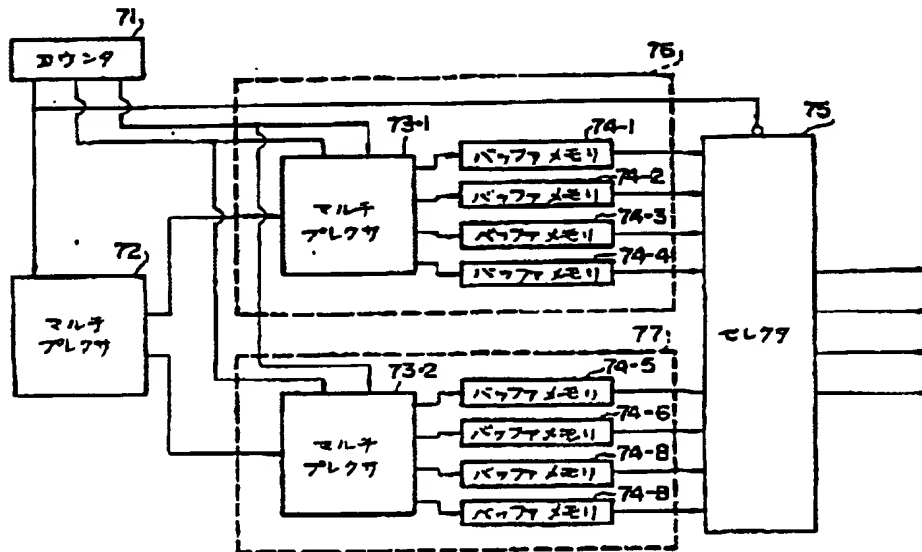
第 2 図



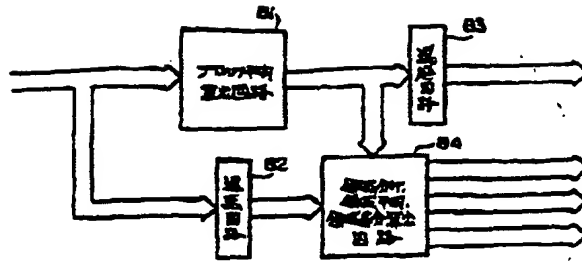
第5図



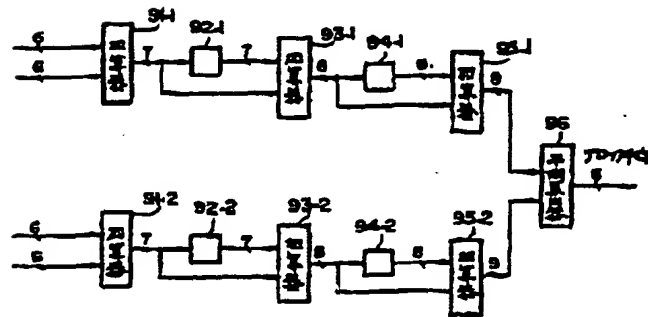
第6図



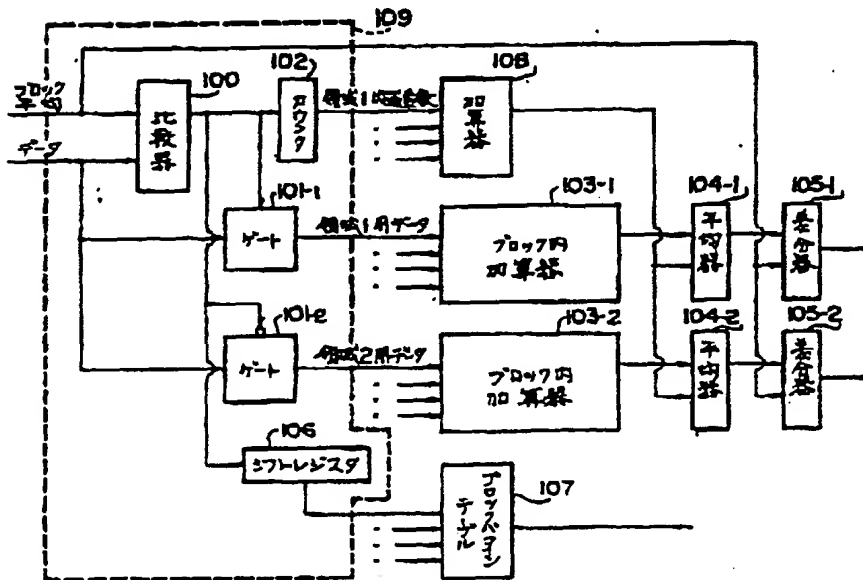
第7図



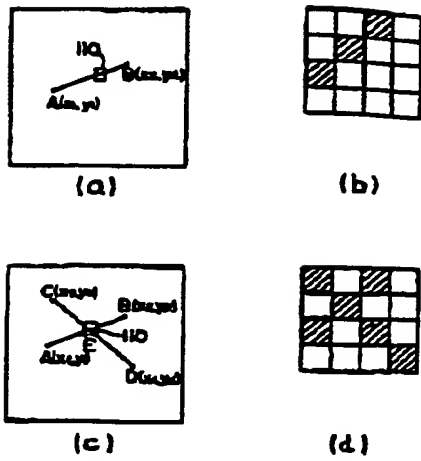
第 8 図



第 9 図



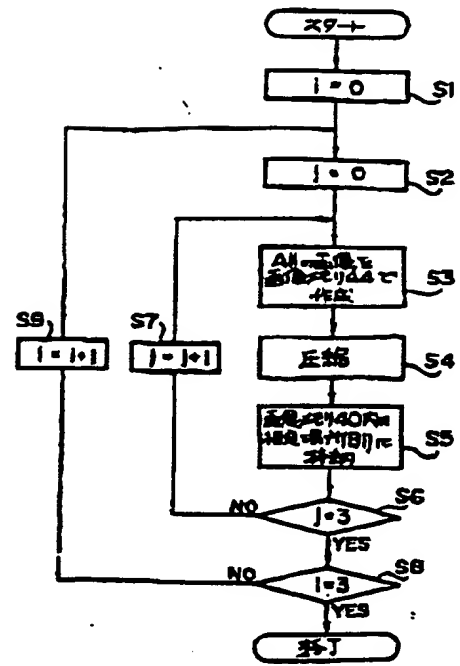
第 10 図



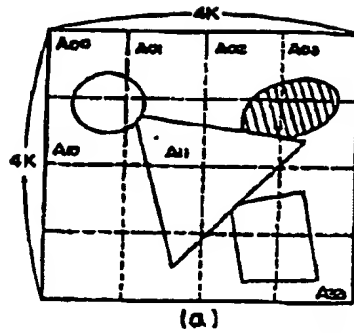
第11図



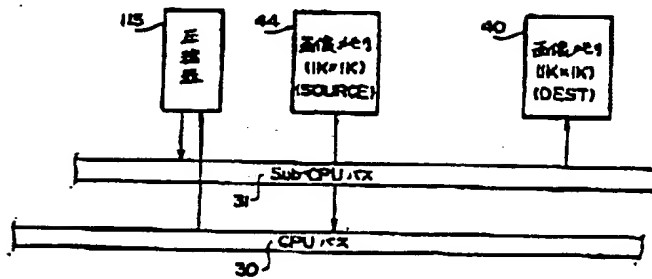
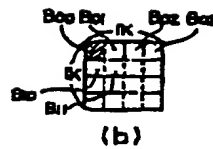
第12図



第15図



第13図



第14図